DETI

Computação em Larga Escala

General Problems – Algorithmic analysis 3

António Rui Borges

Summary

- *Sorting a sequence of integers*
  - *Rationale*
  - *Alternatives*
  - *Algorithm*

DETI

Sorting a sequence of integers - 1

The approach to be followed stems from the fact that it is less complex to sort a somewhat presorted sequence than a totally random one.

Thus, suppose the sequence of N values `val`, with $N = 2^k$ and $k \geq 1$, is such that both its halves, from 0 to $N/2-1$ and from $N/2$ to $N-1$, respectively, are already sorted. In order to sort the whole sequence one may use merge sorting.

Two alternatives are possible for sorting in ascending order

- both halves are sorted in ascending order (*standard merge sorting*)
- first half is sorted in ascending order and second half in descending order (*bitonic sorting*).

Another interesting feature of this approach is that the sorting of both halves is independent from one another and can be done in parallel.

DEEI

Sorting a sequence of integers - 2

Standard merge sorting

Example using an 8-valued sequence

```
1  5  6  9    2  4  7  8   --- initial situation
1  4  6  8    2  5  7  9   --- iteration 1 (4 CAPS): first 2 values positioned
1  2  5  7    4  6  8  9   --- iteration 2 (3 CAPS): another 2 values positioned
1  2  4  6    5  7  8  9   --- iteration 3 (2 CAPS): another 2 values positioned
1  2  4  5    6  7  8  9   --- iteration 4 (1 CAPS): last 2 values positioned
```

DETI

```
for (m = 0; m < N/2; m++)
  for (n = 0; (m + n) < N/2; n++)
    CAPS (val[m+n], val[N/2+n]);
```

`CAPS` stands for *compare and possible swap the value*

The merge operation costs $N(N+2)/8$ `CAPS`, when N is even.

Sorting a sequence of integers - 3

Bitonic sorting

Example using an 8-valued sequence

DETI

```
4  6  8  9    7  5  2  1   --- initial situation
4  5  2  1    7  6  8  9   --- iteration 1 (4 CAPS)
2  1  4  5    7  6  8  9   --- iteration 2 (4 CAPS)
1  2  4  5    6  7  8  9   --- iteration 3 (4 CAPS)
```

Sorting a sequence of integers - 4

```
v = N >> 1;
nL = 1;
for (m = 0; m < k; m++)
{ n = 0;
  u = 0;
  while (n < nL)
  { for (t = 0; t < v; t++)
      CAPS (val[t+u], val[t+u+v]);
    u += (v << 1);
    n += 1;
  }
  v >>= 1;
  nL <<= 1;
}
```

DEI

`CAPS` stands for *compare and possible swap the value*

The merge operation costs $kN/2$ `CAPS`, when N is even.